Algorithms for Fast Digital Straight Segments Union

Isabelle Sivignon

Algorithms for Fast Digital Straight Segments Union

Isabelle Sivignon*

gipsa-lab, CNRS, UMR 5216, F-38420, France
isabelle.sivignon@gipsa-lab.grenoble-inp.fr

Abstract. Given two Digital Straight Segments (DSS for short) of known minimal characteristics, we investigate the union of these DSSs: is it still a DSS ? If yes, what are its minimal characteristics ? We show that the problem is actually easy and can be solved in, at worst, logarithmic time using a state-of-the-art algorithm. We moreover propose a new algorithm of logarithmic worst-case complexity based on arithmetical properties. But when the two DSSs are connected, the time complexity of this algorithm is lowered to $\mathcal{O}(1)$ and experiments show that it outperforms the state-of-the art one in any case.

Keywords: Digital geometry, Union, Digital straight segment

1 Introduction

Digital Straight Lines (DSL) and Digital Straight Segments (DSS) have been used for many years in many pattern recognition applications involving digital curves. Whether it be for polygonal approximation or to design efficient and precise geometric estimators, a basic task is the so-called DSS recognition problem: given a set of pixels, decide whether this set is a DSS and compute its characteristics. Many linear-in-time algorithms have been proposed to solve this problem through the years. Furthermore, Constructive Solid Geometry-like operations have been considered for these objects: the intersection of two DSL has been studied in [15, 16, 8, 5], algorithms for the fast computation of subsegments were described in [9, 17]. Surprisingly enough, the union of DSSs has not yet been studied. The problem was raised in [2] in the context of parallel recognition of DSSs along digital contours. The recognition step was followed by a merging step where the problem of DSSs union appeared. In this work, we show how to solve this problem, both using state-of-the-art algorithm, and proposing a new and faster algorithm.

* This work was partially founded by the French *Agence Nationale de la Recherche* (Grant agreement ANR-11-BS02-009)

2 General considerations

2.1 Preliminary definitions

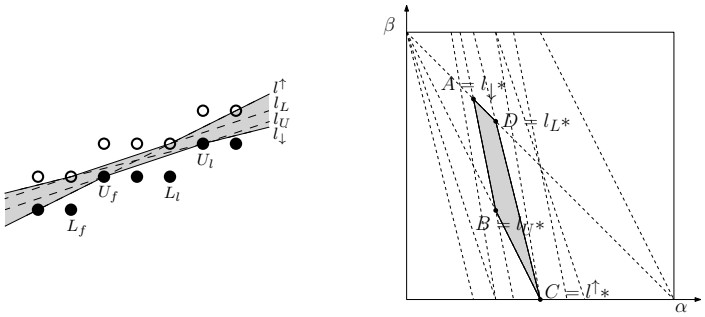
A *Digital Straight Line* (DSL for short) of integer characteristics (a, b, μ) is the infinite set of digital points $(x, y) \in \mathbb{Z}^2$ such that $0 \leq ax - by + \mu < max(|a|, |b|)$ [4]. These DSL are 8-connected and often called *naive*. The fraction $\frac{a}{b}$ is the slope of the DSL, and $\frac{\mu}{b}$ is the shift at the origin. In the following, without loss of generality, we assume that $0 \leq a \leq b$. The remainder of a DSL of characteristics (a, b, μ) for a given digital point (x, y) is the value $ax - by + \mu$. The *upper (resp. lower) leaning line* of a DSL is the straight line $ax - by + \mu = 0$ (resp. $ax - by + \mu = b - 1$). Upper (resp. lower) leaning points are the digital points of the DSL lying on the upper (resp. lower) leaning lines. A *Digital Straight Segment* (DSS) is a finite 8-connected part of a DSL.

If we consider the digitisation process related to this DSL definition, the points of the DSL $\mathbf{L}$ of parameters (a, b, μ) are simply the grid points (x, y) lying below or on the straight line $l : ax - by + \mu = 0$ (Object Boundary Quantization), and such that the points $(x, y+1)$ lie above l. Otherwise said, line l separates the points X of the DSL from the points $X + (0, 1)$ [14], and is called *separating line*. More generally, for an arbitrary set of digital points X, the separating lines are the lines that separate the points X from the points $X + (0, 1)$. In other words, the separating lines separate the upper convex hull of X from the lower convex hull of $X + (0, 1)$. Computing the set of separating lines of two polygons is a very classical problem of computational geometry. It is well known that specific lines called critical support lines can be defined: there are the separating lines passing through a point of each polygon boundary. Critical support points are the points of the polygons belonging to critical support lines [12].

All the separating lines of a DSL have the same slope, but this is not true for arbitrary sets of digital points. The *minimal characteristics* of a set of digital points X are the characteristics of the separating line of minimal b and minimal μ. The set of separating lines of a DSS is well known, and the critical support points are exactly defined by the DSS leaning points: they define the minimal characteristics of the DSS.

The set of separating lines of a set of points X can also conveniently be defined in a dual space, also called parameter space. In this space a straight line $l : \alpha x - y + \beta = 0$ is represented by the 2D point (α, β). Given a set of digital points X, a line $l : \alpha x - y + \beta = 0$ is a separating line if and only if for all $(x, y) \in X, 0 \leq \alpha x - y + \beta < 1$. This definition is strictly equivalent to the one given previously. The preimage of X is the representation of its separating lines in the dual space and is defined as $\mathcal{P}(X) = \{(\alpha, \beta), 0 \leq \alpha \leq 1, 0 \leq \beta \leq 1 | \ \forall (x, y) \in X, 0 \leq \alpha x - y + \beta < 1\}$. The set of separating lines of a set of pixels is an open set in the digital space, but it is a convex polygon in the dual space. In this work, this dual space will not be used explicitly in the algorithms, but we will see that this representation is convenient in some proofs. Moreover, the arrangement of all the constraints for any pixel (x, y) with $y \leq x \leq n$ is called Farey Fan [10] of order n: each cell of this arrangement is the preimage of a DSS

of length n. Figure 1 is an illustration of the separating lines of a DSS, both in the digital space and in the dual space: they separate the points X of the DSS in black, from the points $X + (0, 1)$ in white. Note that the edges of the preimage of a DSS are exactly supported by the dual representation of its leaning points, or equivalently its critical support points.



(a) (b)

Fig. 1. (a) DSS of minimal characteristics $(1, 3, 1)$ with its leaning points U_f, U_l, L_f, L_l. (b) Each vertex of the preimage maps to a straight line in the digital space. The vertex $B(\frac{1}{3}, \frac{1}{3})$ maps to the upper leaning line, the characteristics of which are the minimal characteristics of the DSS.

2.2 Setting the problem and useful properties

Consider now the following problem :

Problem 1. Given two DSSs $S_1 = [P_1Q_1]$ and $S_2 = [P_2Q_2]$ of known minimal characteristics, decide if there exists a DSL containing both S_1 and S_2. If yes, compute the minimal characteristics of $S_1 \cup S_2$.

If S_1 and S_2 do not belong to the same octant, then it is easy to conclude that there is no DSL containing both S_1 and S_2. Thus, in the following S_1 and S_2 belong to the first octant, i.e. we have $S_1(a_1, b_1, \mu_1)$ and $S_2(a_2, b_2, \mu_2)$ with $0 \leq a_1 \leq b_1$ and $0 \leq a_2 \leq b_2$. We denote by $r_i(x, y) = a_i x - b_i y + \mu_i, i \in \{1, 2\}$ the remainder function of each DSS.

In what follows, we moreover suppose that the leaning points of S_1 and S_2 are known as input data. This is not a strong requirement since the most efficient recognition algorithms actually compute this data on the fly.

By convention, we also suppose that the abscissa of Q_2 is greater than the abscissa of Q_1. Note that we make no assumption on the connectivity of S_1 and S_2: the abscissa of P_2 can be lower than, equal to or greater than the abscissa of Q_1. If the abscissa of P_2 is lower than the abscissa of P_1, then the

problem is trivial since S_1 is either a subsegment of S_2 or the union is impossible. Consequently, we also assume that the abscissa of P_2 is greater than the abscissa of P_1.

The first part of Problem 1 consists in deciding if there exists a separating line for the set $S_1 \cup S_2$: we will say that the union is *possible* in this case. If so, then among all the separating lines, the final goal is to find the one with minimal characteristics.

Property 1. The preimage of $S_1 \cup S_2$ is equal to the intersection between the preimages of S_1 and S_2.

Proof. The proof is straightforward since the lines that are separating for $S_1 \cup S_2$ are the ones that are separating for S_1 and S_2.

Corollary 1. *The critical support points of the set of separating lines of $S_1 \cup S_2$ are either upper leaning points or lower leaning points translated by $(0, 1)$ of S_1 and S_2. Thus, to compute the set of separating lines of $S_1 \cup S_2$, it is enough to update the set of separating lines of S_1 with the leaning points of S_2 (or conversely).*

Proof. The critical support points are, in the dual space, lines supporting the edges of the preimage. From Property 1, the lines supportting the edges of $\mathcal{P}(S_1 \cup S_2)$ are lines supporting the edges of $\mathcal{P}(S_1)$ or $\mathcal{P}(S_2)$. However, since S_1 and S_2 are DSSs, the edges of their preimages are supported by the dual representation of either upper leaning points or lower leaning points translated by $(0, 1)$.

3 Fast Union of DSSs: an arithmetical algorithm

3.1 Fast computation of the set of separating lines

A first straightforward solution to compute the set of separating lines of $S_1 \cup S_2$ is to use the state-of-the-art algorithm of O'Rourke [11], re-interpreted in the digital space by Roussillon [14]. Whether it be in the dual space or in the digital space, these algorithms update the critical support points iteratively for each point added. Since at most four points have to be considered in our case, the algorithm is already quite efficient compared to the classical arithmetical recognition algorithm for instance. However, we propose an algorithm that is both faster and simpler to implement, in the spirit of the arithmetical recognition algorithm.

The idea is the following: if we know that the slopes of the separating lines of $S_1 \cup S_2$ are greater/lower than the slopes (given by the minimal characteristics) of S_1 and S_2 respectively, then we can conclude that some leaning points of S_1 or S_2 cannot be critical support points for $S_1 \cup S_2$.

Property 2. Let S_1 be a DSS of minimal characteristics (a_1, b_1, μ_1). Let L_{1f}, L_{1l}, U_{1f}, U_{1l} be its first and last, lower and upper leaning points. If all the separating lines of $S_1 \cup S_2$ have a slope greater (res. lower) than $\frac{a_1}{b_1}$, then U_{1l} (resp. U_{1f}) and L_{1f} (resp. L_{1l}) are not critical support points for $S_1 \cup S_2$.

Proof. If all the separating lines of $S_1 \cup S_2$ have a slope lower than $\frac{a_1}{b_1}$, then, in the dual space and from Property 1, $\mathcal{P}(S_1 \cup S_2)$ is a subpart of the triangle defined by the vertices ABD (see Figure 1(b)). In particular, the edges $[BC]$ and $[DC]$ of $\mathcal{P}(S_1)$ supported by U^*_{1f} and L^*_{1l} respectively cannot be edges of $\mathcal{P}(S_1 \cup S_2)$. Therefore, the leaning points U_{1f} and L_{1l} are not critical support points for $S_1 \cup S_2$. The proof is the same if we suppose that the separating lines all have a slope greater than $\frac{a_1}{b_1}$.

Note that if S_1 has three leaning points only, let's say for instance only one lower leaning point L_1, then setting L_{1l} and L_{1f} to L_1 (L_1 is "duplicated"), the property is also valid. A similar result holds when the leaning points of S_2 are considered. However, guessing the slope of the union can be tricky, and taking into account only the DSS slopes is not enough. For example, it is easy to exhibit cases where the slope of S_2 is greater than the slope of S_1, and the slope of $S_1 \cup S_2$ is nevertheless lower than both the slope of S_1 and the slope of S_2 (see Figure 2(a)). We establish hereafter some properties linking the remainder of the leaning points of S_2 and the slope of the separating lines for $S_1 \cup S_2$ if they exist.

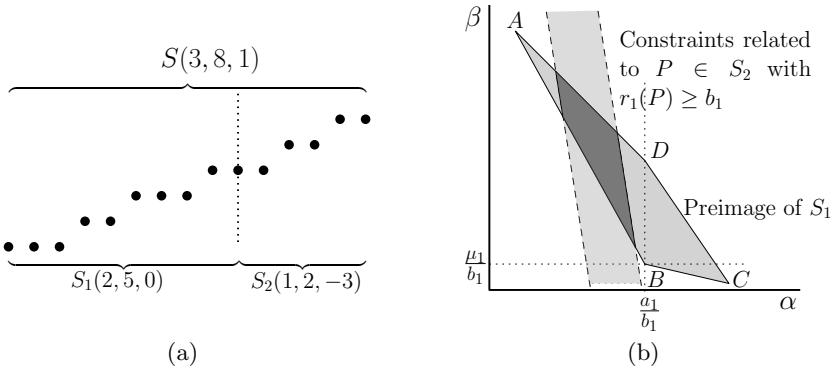


Fig. 2. (a) The slope of S_1 is equal to $\frac{2}{5}$ and lower than the slope of S_2, which is equal to $\frac{1}{2}$. However, the slope of $S_1 \cup S_2$ is equal to $\frac{3}{8}$ and smaller than both. (b) Illustration of Property 4.

Let's start with a very simple consideration.

Property 3. If for all the leaning points P of S_2 (resp. S_1), we have $0 \leq r_1(P) < b_1$ (resp. $0 \leq r_2(P) < b_2$), then there exists a DSL containing $S_1 \cup S_2$ and its minimal characteristics are the one of S_1 (resp. S_2).

Proof. In the dual space, the points B and D of the preimage of S_1 satisfy all the constraints related to the points P defined as above, which ends the proof.

With the following property, we investigate the other cases.

Property 4. Let P be a leaning point of S_2.

- if $\mathbf{r_1(P) \geq b_1}$, the slope of all the separating lines for $S_1 \cup S_2$, if any, is **lower** than the slope of S_1
- if $\mathbf{r_1(P) < 0}$, the slope of all the separating lines for $S_1 \cup S_2$, if any, is **greater** than the slope of S_1

Let P be a leaning point of S_1.

- if $\mathbf{r_2(P) \geq b_2}$, the slope of all the separating lines for $S_1 \cup S_2$, if any, is **greater** than the slope of S_2
- if $\mathbf{r_2(P) < 0}$, the slope of all the separating lines for $S_1 \cup S_2$, if any, is **lower** than the slope of S_2

Proof. We prove the first item, for a leaning point of S_2 with a remainder greater than or equal to b_1. Proving the other cases is similar. Consider a point $P \in S_2$ such that $r_1(P) \geq b_1$. Let us consider the stripe defined by the constraints related to this point in the dual space. It is very simple to see that the point $(\frac{a_1}{b_1}, \frac{\mu_1}{b_1})$ is above this stripe (see Figure 2(b)). Since P does not belong to S_1, and with the assumptions made in Section 2.2 on the relative position of S_1 and S_2, its abscissa is greater than the abscissas of all the leaning points of S_1. This means that the intersection, if not empty, between the stripe and $\mathcal{P}(S_1)$ lies in the subspace $\alpha < \frac{a_1}{b_1}$.

Table 1 summarises the computation of the four possible critical support points combining Properties 2 and 4. Figure 6 in Appendix illustrates the first line of this table.

	value / remainder	< 0	$\geq b$
$P \in S_2$	$r_1(P)$	$(U_f, L_f) = (U_{1f}, L_{1l})$	$(U_f, L_f) = (U_{1l}, L_{1f})$
$P \in S_1$	$r_2(P)$	$(U_l, L_l) = (U_{2l}, L_{2f})$	$(U_l, L_l) = (U_{2f}, L_{2l})$

Table 1. Possible critical support points according to remainder values

At this point, we have identified four points denoted by U_f, U_l, L_f and L_l, that may be critical support points for $S_1 \cup S_2$. However, they may not be all critical support points. Since the preimage of $S_1 \cup S_2$ is a convex polygon, it has at least three edges and thus, at least three out of the four possible points are indeed critical support points. Property 5 gives a way to decide whether the four points are critical support points or not.

Property 5. U_f and U_l (resp. L_f and L_l) are both critical support points if and only if L_f and L_l (resp. U_f and U_l) belong to the DSL of directional vector $U_l - U_f$ (resp. $L_l - L_f$) and upper (resp. lower) leaning points U_f and U_l (resp. L_f and L_l).

Proof. U_f and U_l are both critical support points is equivalent to say that the straight line (U_fU_l) is separating for $S_1 \cup S_2$. This is also equivalent to the fact that L_f and L_l belong to the DSL as defined in the property statement.

If the four points are not all critical support points, the three critical support points are identified using Property 6.

Property 6. Let U_f, U_l, L_f and L_l be the four possible critical support points for $S_1 \cup S_2$. If they are not all critical support points, then:

- if U_f and U_l are both critical support points then:
  - if the slope of (U_fU_l) is lower than the slope of (L_fL_l), L_f is the third critical support point ;
  - otherwise, L_l is the third critical support point ;
- if L_f and L_l are both critical support points then:
  - if the slope of (L_fL_l) is greater than the slope of (U_fU_l), U_f is the third critical support point ;
  - otherwise, U_l is the third critical support point ;

Proof. We write the proof for the case where U_f and U_l are both critical support points. The other case is similar.

Consider the dual representation of the points U_f and U_l, denoted by U_f^* and U_l^*. By hypothesis, these two lines support edges of $\mathcal{P}(S_1 \cup S_2)$. Consider now the dual representation of the points $L_f + (0, 1)$ and $L_l + (0, 1)$, denoted by L_{f+}^* and L_{l+}^*. The third edge of $\mathcal{P}(S_1 \cup S_2)$ is a segment of either L_{f+}^* or L_{l+}^*. We suppose now that the slope of (U_fU_l) is lower than the slope of (L_fL_l) and illustrate the rest of the proof with Figure 3. Then, the abscissa of point $D = L_{f+}^* \cap L_{l+}^*$ is greater than the abscissa of point $B = U_f^* \cap U_l^*$ (it lies in the light-gray half-space on Figure 3). It is now easy to see that if D is above the line U_f^*, then both L_{f+}^* and L_{l+}^* support edges of $\mathcal{P}(S_1 \cup S_2)$, which is not possible by hypothesis. Then, D is below the line U_f^*, which implies that the third edge of the preimage is a segment of L_{f+}^*, and equivalently L_f is the third critical support point.

3.2 Pulling out the minimal characteristics

In the previous section, we showed how to efficiently compute the three or four critical support points of $S_1 \cup S_2$. These points also define the preimage of $S_1 \cup S_2$. Until now, the results were valid whether the two DSSs were connected or not. In order to compute the minimal characteristics, we have to consider several cases.

Input DSSs are connected We consider here the case where the first point of S_2 is either a point of S_1 or 8-connected to the last point of S_1. In this case, if there exists a DSL containing $S1 \cup S_2$, then $S_1 \cup S_2$ is a DSS of length n, the difference of abscissa between the first point of S_1 and the last point of S_2. As a consequence, $\mathcal{P}(S_1 \cup S_2)$ is a cell of the Farey Fan of order n, with very well-known properties. In particular, the critical support points computed in Section 3.1 are exactly the leaning points of the DSS.

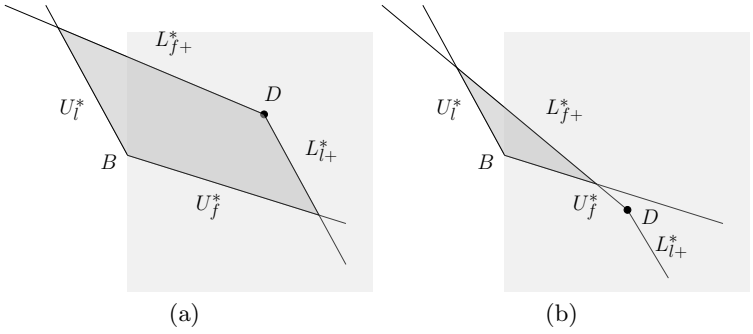


Fig. 3. Illustration of the proof of Property 6.

S_1 last point and S_2 first point have the same ordinate We show that this case is actually as easy as the previous one.

Property 7. If the last point of S_1 and the first point of S_2 have the same ordinate, then $\mathcal{P}(S_1 \cup S_2)$ is a unique cell of the Farey Fan of order n, the difference of abscissa between the last point of S_2 and the first point of $S1$.

Proof. In a DSS, the edges of the preimage are defined by the leaning points only. Actually, the preimage of a DSS is equal to the preimage of its leaning points, all the other points make no contribution. If the last point of S_1 and the first point of S_2 have the same ordinate, then all the missing points between these two points also have this same ordinate. Consequently, they cannot be neither lower nor upper leaning points for any DSL containing $S_1 \cup S_2$. This proves that $\mathcal{P}(S_1 \cup S_2)$ is the same as the preimage of the set of pixels composed of S_1, S_2 and all the missing points between the two. Then, $\mathcal{P}(S_1 \cup S_2)$ is the preimage of a DSS of length n, the difference of abscissa between the last point of S_2 and the first point of $S1$, which is similar to the previous case.

Disconnected DSSs This case is trickier since $\mathcal{P}(S_1 \cup S_2)$ may not be a unique cell but can be a union of adjacent cells of a Farey Fan (see Figure 4 for an example). The characteristics given by the critical points may not be the minimal ones. However, from the critical support points we can easily infer the range of slopes of the separating lines. If we denote s_{low} and s_{up} the minimum and maximum slopes of the separating lines, the slope of the line of minimal characteristics is given by the fraction of smallest denominator between s_{low} and s_{up}. It is finally easy to decide which one of either U_f or U_l is an upper leaning point of the line of minimal characteristics (see Algorithm 1 for more details).

3.3 General algorithm

All the properties presented above are put together to design the fast union algorithm described in Algorithm 1. The algorithm returns the minimal characteristics of $S_1 \cup S_2$ if the union is possible. The result is given as a directional

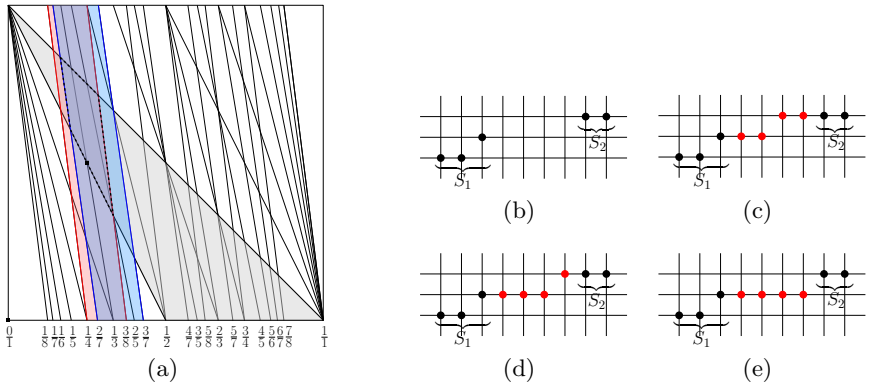


Fig. 4. When S_1 and S_2 are not connected, like the ones depicted in (b), $\mathcal{P}(S_1 \cup S_2)$ may be the union of several cells: in (a), $\mathcal{P}(S_1)$ is depicted in light gray, and the two constraints related to the leaning points of S_2 are depicted in red and blue. The intersection is bordered by a dotted black line: it is composed of three cells, each one being the preimage of a DSS containing $S_1 \cup S_2$, depicted in (c), (d), and (e).

vector (b, a) and an upper leaning point. The algorithm can be decomposed into three main parts. Between line 1 and 2, the four possible critical support points are computed. The function `initCriticalSupportPoint` is the implementation of Table 1 and is detailed in Algorithm 2 presented in Appendix. The string parameter given in input is there to discriminate between the two lines of Table 1. At the same time, easy cases where $S_1 \cup S_2$ has the same minimal characteristics as S_1 or S_2 are treated: in such cases, the variable `inDSL` is set to true by `initCriticalSupportPoint` and we can conclude directly. Then, between lines 3 and 4, the exact critical support points are computed. The function `isSolution`? implements Property 5 and is detailed in Algorithm 3 in Appendix: here, the string parameter tells if the upper leaning points are tested w.r.t the lower leaning points or conversely. The function `thirdPoint` implements Property 6: if the variable `solU` is true, then the first item of the property is concerned, otherwise `solL` is true, and the second item is concerned. The last part, between lines 5 and 7 returns the minimal characteristics of $S_1 \cup S_2$. On line 6, if S_1 and S_2 are connected or if the last point of S_1 and the first point of S_2 have the same ordinate, the result is straightforward from the critical support points. Otherwise, as explained in Section 3.2 the line of minimal characteristics is computed among all the separating lines. Function `minimalSlope` is then called. It can be implemented in several way, using for instance the decomposition into continued fractions, or, like in [16]-Algorithm 3 (see also [7, 6]) using the Stern-Brocot tree.

3.4 Complexity analysis

Lemma 1. *The complexity of Algorithm 1 is $\mathcal{O}(1)$ when S_1 and S_2 are connected or when the last point of S_1 and the first point of S_2 have the same ordinate. Its*

Algorithm 1: `FastArithmeticalDSSUnion(DSS` S_1`, DSS` S_2`)`

U_f, U_l, L_f, L_l critical support points of $S_1 \cup S_2$;
boolean inDSL $\leftarrow$ false; boolean solU, solL;
connected $\leftarrow$ true if S_1 and S_2 are connected or the last point of S_1 and the first point of S_2 have the same ordinate, false otherwise;

1 (U_l, L_l, inDSL) $\leftarrow$ `initCriticalSupportPoints(`S_1, S_2, *"after"*`)`
if inDSL $=$ *true* **then return** DSS(a_1,b_1,U_{1f})
else
  (U_f, L_f, inDSL) $\leftarrow$ `initCriticalSupportPoints(`S_2, S_1, *"before"*`)`
  if inDSL $=$ *true* **then return** DSS(a_2,b_2,U_{2l})
  else
    `// Four possible critical support points are known`
3   (solL,a_L, b_L) $\leftarrow$ `isSolution`? (L_f,L_l,U_f,U_l,"lower")
    (solU,a_U, b_U) $\leftarrow$ `isSolution`? (U_f,U_l,L_f,L_l,"upper")
    if solU $=$ *false and* solL $=$ *false* **then**
      return DSS(0,0,Point(0,0)) `// Union of` S_1 `and` S_2 `is not possible`
    else
      if solU $=$ *false or* solL $=$ *false* **then**
        `// Three points only are critical`
        if solU $=$ *true* **then** $(a, b) \leftarrow (a_U, b_U)$ **else** $(a, b) \leftarrow (a_L, b_L)$
        (U_f,U_l,L_f,L_l) $=$ `thirdPoint` (U_f,U_l,L_f,L_l,solU,solL)
4   **else** $(a, b) \leftarrow (a_U, b_U)$ `// The four points are critical`
    `// At this point, the exact critical support points are known`
5   **if** *connected* $=$ *true* **then**
      return DSS(a,b,U_f)
    else
6     $(a, b) \leftarrow$ `minimalSlope` (U_f,U_l,L_f,L_l)
      if $U_l = U_f$ **then** $U \leftarrow U_f$
      else
        if *slope*$((U_f\ U_l)) > \frac{a}{b}$ **then** $U \leftarrow U_f$ **else** $U \leftarrow U_l$
7   **return** DSS(a,b,U)

complexity is $\mathcal{O}(log(n))$ *otherwise, where* n *is the difference of abscissa between the last point of* S_2 *and the first point of* S_1.

Proof. If we assume a computing model where the standard arithmetic operations are done in constant time, then all the operations from line 1 to line 5 are also done in constant time. Whichever the algorithm chosen, the function `minimalSlope` on line 6 always requires, in a more or less direct way, the computation of the continued fractions of two fractions $\frac{p}{q}$ with $p \leq q \leq n$, and n is the difference of abscissa between the last point of S_2 and the first point of S_1. This is done in $\mathcal{O}(log(n))$ time (see [6] for instance).

4 Experimental results

Algorithm 1 was implemented in C++ using the open-source library DGtal [1]. We refer to it as the `FastArithmetical` algorithm in the following. We compare our algorithm with two other ones. The first one is the well-known arithmetical recognition algorithm [4], implemented in DGtal (called `Arithmetical` algorithm in what follows). As stated at the beginning of Section 3.1, the algorithm of O'Rourke [11] can be used to compute the set of separating lines. It was implemented in DGtal by T. Roussillon as the `StabbingLine` algorithm. The `Arithmetical` algorithm works only when the two DSSs are connected and is used as follows : the minimal characteristics are initialised with the ones of S_1, and updated as the points of S_2 are added one by one. Concerning the `StabbingLine` algorithm, the preimage is initialised with the one of S_1 and updated as the leaning points of S_2 are added (Corollary 1). The result is the set of critical support points of $S_1 \cup S_2$.

The experimental setup is the following:

- a DSL of characteristics (a, b, μ) is picked up at random ;
- the abscissas x_1 and x_2 of the first and last points of S_1 are randomly selected ;
- the abscissa x_3 of the first point of S_2 is either equal to x_2+1 in the connected case, or randomly selected and greater than x_2 in the disconnected one ;
- the abscissa of the last point is set at a fixed distance from x_3 ;

Two parameters govern this setup : `maxb` is the maximal value of b ; `valX` is the length of S_2. b is randomly picked in the interval $[1, \texttt{maxb}]$, a is drawn in the interval $[1, b]$ and such that a and b are relatively prime, and μ in the interval $[0, 2\texttt{maxb}]$. The value of x_1 is drawn in the interval $[0, \texttt{maxb}]$. The length of S_1 (*i.e.* $x_2 - x_1$) is randomly selected in the interval $[\texttt{valX}, 2\texttt{valX}]$, so that S_1 is always longer than S_2. In our test, `maxb` is set to 1000 and `valX` varies from 10 to 2`maxb`. For each value of `valX`, 2000 pairs of values (a, b) are drawn. For each of them, 5 different values of μ are picked up, and then 10 different values of x_1 are tested, for a total of 10^5 tests.

When the two DSSs are connected, the first test we perform consists in verifying that the three algorithms actually compute the same minimal characteristics. Then, the performances in terms of computation time are compared. Figure 4 shows the results (logarithmic scale for both axis): the x-axis represents the value of `valX`, the y-axis is the mean CPU computation time for a pair of DSSs, and for each algorithm.

First we can observe that the experimental behaviour of `FastArithmetical` algorithm confirms the constant-time complexity. Unsurprisingly, the `Arithmetical` algorithm has a linear-time complexity. Concerning the `StabbingLine` algorithm, its performances are slightly worse than the `FastArithmetical` algorithm, and a slight increase of the mean computation time is observed for larger DSS lengths: this is due to the fact that a post-treatment has to be done on the result returned by this algorithm in order to compute the minimal characteristics. This post-treatment involves a *gcd* computation, which explains the plot.

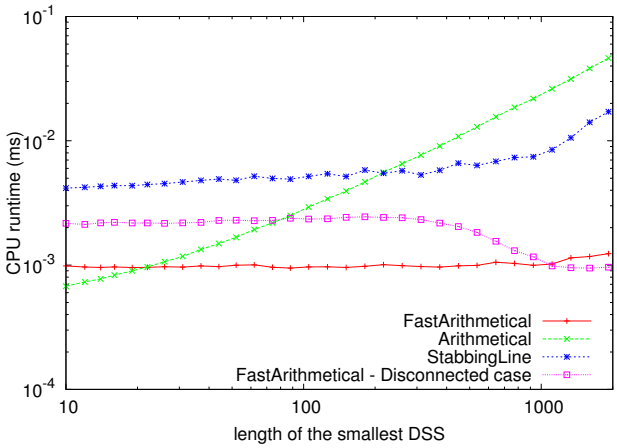


Fig. 5. Experimental results

However, the main information is that the `FastArithmetical` algorithm gets faster than the classical `Arithmetical` one when the length of the smallest DSS is greater than 20. In comparison, the `StabbingLine` algorithm becomes faster for lengths greater than 200 only. This means that what could appear as a small gain on a constant term in comparison to the `StabbingLine` algorithm makes the `FastArithmetical` relevant in practice compared to the `Arithmetical` algorithm. Last, the `FastArithmetical` remains faster than the `StabbingLine` algorithm even when the two DSSs are not connected. The slight decrease of the mean computation time for long DSSs is related to the fact that the longer the DSSs, the more easy cases appear.

5 Conclusion

In this work, we have shown that the union of two DSSs can be very efficiently computed since it is enough to "update" the minimal characteristics of the first segment with the leaning points of the second one. To do so, we have demonstrated that a state-of-the-art algorithm - the stabbing line algorithm - can be used to compute the union in logarithmic time. Moreover, we have exhibited a number of simple arithmetical properties to design an even faster algorithm. This algorithm runs in $\mathcal{O}(log(n))$ worst-time complexity, and $\mathcal{O}(1)$ for easy cases and the experiments have shown that the implementation concretises this complexity.

Now, an interesting question remains: what if the union is not possible ? Can we measure the "distance" between the two DSSs ? A solution would be to consider "thicker" DSSs and to compute the thickness necessary for the union to be possible. This problem seems actually to be very close the the blurred DSS recognition algorithms [3, 13], and this trail seems worthy to be followed.

References

1. DGtal: Digital Geometry Tools and Algorithms Library. `http://libdgtal.org`
2. Damiand, G., Coeurjolly, D.: A generic and parallel algorithm for 2d digital curve polygonal approximation. Journal of Real-Time Image Processing (JRTIP) 6(3), 145–157 (2011)
3. Debled-Rennesson, I., Feschet, F., Rouyer-Degli, J.: Optimal blurred segments decomposition of noisy shapes in linear time. Computers & Graphics 30(1), 30 – 36 (2006)
4. Debled-Rennesson, I., Reveillès, J.P.: A linear algorithm for segmentation of digital curves. Inter. Jour. of Pattern Recog. and Art. Intell. 9(6), 635–662 (1995)
5. Debled-Rennesson, I., Reveillès, J.P.: A new approach to digital planes. In: SPIE - Vision Geometry III (1994)
6. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics. Addisson-Wesley (1994)
7. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestor. SIAM Journal on Computing 13(2), 338–355 (1984)
8. Jacob, M.A.: Applications quasi-affines. Ph.D. thesis, Université Louis Pasteur, Strasbourg, France (1993)
9. Lachaud, J.O., Said, M.: Two efficient algorithms for computing the characteristics of a subsegment of a digital straight line. Discrete Applied Mathematics 161(15), 2293 – 2315 (2013)
10. McIlroy, M.D.: A note on discrete representation of lines. AT&T Technical Journal 64(2), 481–490 (1985)
11. O'Rourke, J.: An on-line algorithm for fitting straight lines between data ranges. Commun. ACM 24(9), 574–578 (1981)
12. O'Rourke, J.: Computational Geometry in C. Cambridge University Press (1998)
13. Roussillon, T., Tougne, L., Sivignon, I.: Computation of binary objects sides number using discrete geometry, application to automatic pebbles shape analysis. In: Int. Conf. on Image Analysis and Processing. pp. 763–768 (2007)
14. Roussillon, T.: Algorithmes d'extraction de modèles géométriques discrets pour la représentation robuste des formes. Ph.D. thesis, Université Lumière Lyon 2 (2009)
15. Said, M., Lachaud, J.O., Feschet, F.: Multiscale analysis of digital segments by intersection of 2d digital lines. In: ICPR 2010. pp. 4097–4100 (2010)
16. Sivignon, I., Dupont, F., Chassery, J.M.: Digital intersections : minimal carrier, connectiviy and periodicity properties. Graphical Models 66(4), 226–244 (2004)
17. Sivignon, I.: Walking in the farey fan to compute the characteristics of a discrete straight line subsegment. In: Discrete Geometry for Computer Imagery, Lect. Notes on Comp. Sci., vol. 7749, pp. 23–34 (2013)

Appendix

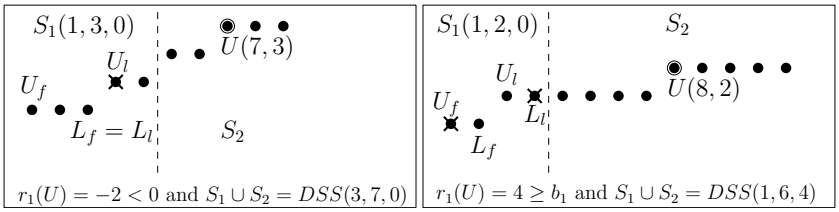


Fig. 6. Illustration of the first line of Table 1: the leaning points of S_1 marked with a cross cannot be critical support points if the point U of S_2 is added.

Algorithm 2: `InitCriticalSupportPoints(DSS` S`, DSS` S'`, string` *position*)

U_f, U_l, L_f, L_l the leaning points of S, r remainder function of S
inDSL a boolean; inDSL $\leftarrow$ true
foreach *leaning point* P *of* S' *and if* inDSL $=$ *true* **do**
  if $r(P) < 0$ **then**
    if position $=$ *"after"* **then** $U \leftarrow U_f$, $L \leftarrow L_l$ **else** $U \leftarrow U_l$, $U \leftarrow U_f$
    inDSL $\leftarrow$ false
  else
    if $r(P) \geq b$ **then**
      if position $=$ *"after"* **then** $U \leftarrow U_l$, $L \leftarrow L_f$ **else** $U \leftarrow U_f$, $L \leftarrow L_l$
      inDSL $\leftarrow$ false
    else
      if $r(P) = 0$ **then** $U \leftarrow$ P
      if $r(P) = b - 1$ **then** $L \leftarrow$ P
end
return (U,L,inDSL)

Algorithm 3: `isSolution?(`P_f`,`P_l`,`Q_f`,`Q_l`, string` *type*)

// compute the characteristics defined by the points P_f and P_l
$a \leftarrow P_l.y - P_f.y$, $b \leftarrow P_l.x - P_f.x$
if type $=$ *"lower"* **then** $\mu \leftarrow b - 1 - aP_f.x + bP_f.y$
else $\mu \leftarrow -aP_f.x + bP_f.y$
// check the position of Q_f and Q_l w.r.t these characteristics
Let $r(Q) = aQ.x - bQ.y + \mu$
if $0 \leq r(Q_f) < b$ *and* $0 \leq r(Q_l) < b$ **then return** (true,a, b, μ)
else
  return (false,a, b, μ)